(12)特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2004 年3 月25 日 (25.03.2004)

PCT

(10) 国際公開番号
WO 2004/025468 A1

(51) 国際特許分類[7]: G06F 11/00, 9/32, 9/34, H03K 19/177

(21) 国際出願番号: PCT/JP2002/009437

(22) 国際出願日: 2002 年9 月13 日 (13.09.2002)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(71) 出願人 (米国を除く全ての指定国について): 株式会社 日立製作所 (HITACHI, LTD.) [JP/JP]; 〒101-8010 東京都 千代田区 神田駿河台四丁目６番地 Tokyo (JP).

(72) 発明者; および

(75) 発明者/出願人 (米国についてのみ): 田中 博志 (TANAKA,Hiroshi) [JP/JP]; 〒185-8601 東京都 国分寺市 東恋ヶ窪一丁目２８０番地 株式会社日立製作所 中央研究所内 Tokyo (JP).

(74) 代理人: 作田 康夫 (SAKUTA,Yasuo); 〒100-8220 東京都 千代田区 丸の内一丁目５番１号 株式会社日立製作所内 Tokyo (JP).

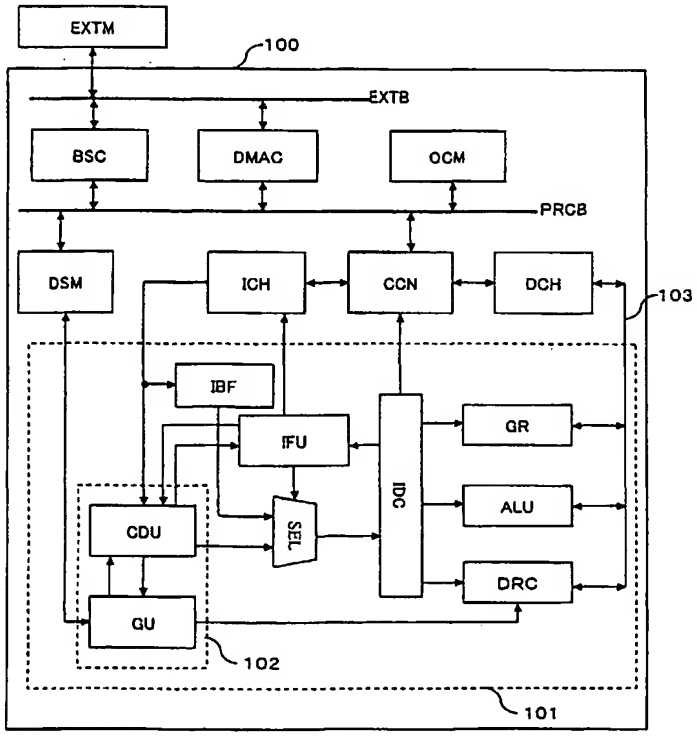
(81) 指定国 (国内): CN, JP, KR, US.

(84) 指定国 (広域): ヨーロッパ特許 (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR).

添付公開書類:
— 国際調査報告書

(54) Title: SEMICONDUCTOR DEVICE

(54) 発明の名称: 半導体装置



(57) Abstract: A semiconductor device having a circuit which can be rewritten dynamically and a function for maintaining software compatibility in spite of the dynamically rewritable configuration. Simultaneously with software execution, data for rewriting a dynamically rewritable circuit and driver software for operating this are automatically generated and the latter part of an original program is replaced by the generated data. Thus, by maintaining software compatibility and diverting an existing software resource, it is possible to use the same software in various devices.

[続葉有]

WO 2004/025468 A1

(57) 要約:

動的書き換え可能な回路を搭載した半導体装置において、動的書き換え可能な回路の構成にかかわらず、ソフトウェアの互換性を維持する機構を提供する。

ソフトウェアが実行されると同時に、動的書き換え可能な回路を書き換えるデータ及びこれを動作させるドライバ・ソフトウェアを自動生成し、元のプログラムを途中から置き換える。

このように、ソフトウェアの互換性を維持することで、既存のソフトウェア資産を流用しつつ様々な機器で同じソフトウェアを用いることができる。

明 細 書

半導体装置

技術分野

本発明は、動的に書き換え可能な回路を搭載した半導体装置の構成と、その利用方法に関する。

背景技術

近年、情報処理機器の普及と高性能化に伴い、様々なアプリケーションが登場しており、これらアプリケーションはソフトウェアで記述され、汎用プロセッサで実行される形態が主流となっている。ところが、アプリケーションの中には汎用プロセッサよりも高度な演算処理能力を要求するものもあり、プロセッサにはより一層の処理能力の向上が要求されるようになっている。

このため、汎用プロセッサに加えて特定のアプリケーションに特化した専用回路を１チップに搭載することにより、処理能力を向上させる例がある。さらに、この専用回路を動的に書き換え可能な回路(Dynamically Reconfigurable Circuit、以下、ＤＲＣと称する）により構成する例が、特開平１０－４３４５号公報や特開平１０－３３５４６２号公報に開示されている。

かかる先行技術では、ＤＲＣ書き換えデータをアプリケーションのソフトウェア作成時にあらかじめ作成しておく。ＤＲＣ書き換えデータによってＤＲＣを書き換えることによって、ＤＲＣは特定のアプリケーションのための専用回路として機能するようになる。汎用プロセッサで実行されるソフトウェアには、ＤＲＣ書き換えデータと書き換え命令とが

含まれている。

これにより、汎用プロセッサはアプリケーションの実行中にＤＲＣを書き換えて専用回路として機能させることにより、その処理能力を向上させることができる。

本願の発明者らは、以上の先行技術のようにＤＲＣ書き換えデータとドライバ・ソフトウェアとを利用するＤＲＣとチップ構成にあわせてあらかじめ用意し、ＤＲＣ書き換え命令とＤＲＣ書き換えデータをソフトウェア内に記述する構成では、ＤＲＣの構成が異なるチップではそのソフトウェアの実行ができなくなってしまうという問題があることに気づいた。このことは、ＤＲＣの構成により適用できるソフトウェアの範囲が制限を受けることとなり、同じ命令セットを持つプロセッサソフトウェアでありながら、ＤＲＣの構成が異なるためにソフトウェアが利用できないという事態を生じさせることになる。

本発明の目的は、ＤＲＣを利用して処理能力を向上させながらも、ＤＲＣの構成によらずソフトウェアの互換性を確保できる半導体集積回路装置を提供することである。

発明の開示

本願において開示される発明のうち代表的なものの概要を簡単に説明すれば下記の通りである。

演算命令を含むソフトウェアを実行する半導体装置であって、複数の演算セルと複数のレジスタセルとを含み、演算セルが実行する演算種と複数の演算セル及び複数のレジスタセル間の配線接続とを設定可能な演算回路と、ソフトウェアに基づき、演算セルの演算種及び配線接続を設定する設定データと、演算回路を用いてソフトウェアと等価な処理を行うためのドライバ・ソフトウェアとを生成する制御回路とを有する。

ここで演算種とは、セル演算の実行可能な論理和、論理積、排他的論理和といった論理演算、加減乗除といった算術演算、比較演算が含まれる。このような構成により、半導体装置上でドライバ・ソフトウェアを生成することが可能になり、ソフトウェアの互換性が確保できる。さらに、ソフトウェアの実行中にドライバ・ソフトウェアを作成することにより、ドライバ・ソフトウェアを作成するオーバーヘッドをユーザが意識することなく、演算回路を用いた高速処理が可能になる。

また、演算命令を含むソフトウェアを実行する半導体装置であって、レジスタと、演算器と、複数の演算セルと複数のレジスタセルとを含み、演算セルが実行する演算種と複数の演算セル及び複数のレジスタセル間の配線接続とを設定可能な演算回路と、ソフトウェアを格納する第１メモリ領域と、演算回路を用いてソフトウェアと等価な処理を行うためのドライバ・ソフトウェアを格納する第２メモリ領域と、実行するソフトウェアを制御する制御回路とを有し、ソフトウェアの処理はｎ回繰り返され、第１回から第ｉ回（ｉ＜ｎ）までの処理は、レジスタと演算器とを用いて第１メモリから読み出されたソフトウェアを実行することによって行われ、制御回路は第ｉ回の処理を受けて、実行するソフトウェアをドライバ・ソフトウェアに切り換えることにより、第ｉ＋１回から第ｎ回までの処理は、演算回路を用いて第２メモリ領域から読み出されたドライバ・ソフトウェアを実行することによって行われる。このようなソフトウェアとドライバ・ソフトウェアとを別のメモリ領域に記憶し、制御回路がソフトウェアとドライバ・ソフトウェアとを切り換える構成により、ソフトウェアの互換性が確保できる。

このような構成は、特に複数回繰り返されるソフトウェア（例えば、ループを形成しているソフトウェア）に有効であり、このようなループは画像処理や音声処理でよく現れる処理である。

図面の簡単な説明

第１図は、本発明の半導体集積回路装置の構成を示すブロック図である。

第２図は、ＤＲＣドライバＳＷの生成及び実行のタイミングを示すための図である。

第３図は、ＤＲＣの構成例を示すブロック図である。

第４図は、ＤＲＣの構成要素である入出力レジスタ・セルＩＯＲＣの構成例を示すブロック図である。

第５図は、ＤＲＣの構成要素である演算セルＣＣの構成例を示すブロック図である。

第６図は、書き換え判定ユニットＣＤＵの構成例を示すブロック図である。

第７図は、書き換え判定ユニットＣＤＵの動作を示すフロー図である。

第８図（Ａ）は、通常ＳＷ、第８図（Ｂ）は通常ＳＷからＨＷ／ＳＷ生成ユニットＧＵによって切り出された通常ＳＷ、第８図（Ｃ）はＨＷ／ＳＷ生成ユニットＧＵによって作成されたＤＲＣドライバＳＷ、第８図（Ｄ）はプログラム中の命令の動作一覧である。

第９図は、ＨＷ／ＳＷ生成ユニットＧＵの動作を示すフロー図である。

第１０図は、ＣＤＦＧの例である。

第１１図は、縮約処理を行ったＣＤＦＧの例である。

第１２図は、ＡＬＡＰ（As Late As Possible）アルゴリズムを用いてスケジューリングを行ったＣＤＦＧの例である。

第１３図は、さらにＡＳＡＰ（As Soon As Possible）アルゴリズムを用いてスケジューリングを行ったＣＤＦＧの例である。

第１４図は、本発明の半導体集積回路装置の別の構成を示すブロック

図である。

第１５図は、本発明の半導体集積回路装置のさらに別の構成を示すブロック図である。

発明を実施するための最良の形態

以下、本発明による代表的な実施の形態を図面に従って詳細に説明する。なお、以下においては、同じ参照符号は同じもの、もしくは類似のものを表す。

本発明では、ＤＲＣを用いることなく汎用プロセッサで実行されるアプリケーションソフトウェア（以下、このようなソフトウェアを「通常ＳＷ」と呼ぶ）から、アプリケーションソフトウェアの一部分をＤＲＣで実行させるためのＤＲＣドライバ・ソフトウェア（以下、このようなソフトウェアを「ＤＲＣドライバＳＷ」と呼ぶ）を自動生成し、汎用プロセッサは、通常ＳＷの一部をＤＲＣドライバＳＷに置き換えて実行することで処理能力を向上させる。

通常ＳＷとＤＲＣドライバＳＷとの関係を第８図（Ａ）～（Ｄ）を用いて説明する。第８図（Ａ）の例は高速フーリエ変換のプログラムの一部である。高速フーリエ変換はマルチメディア処理でよく用いられる処理である。第８図（Ａ）のプログラムは汎用プロセッサの命令セットにより記述されている。プログラムの各命令の意味は第８図（Ｄ）に示している。第８図で用いた命令セットは一例であって、本発明が当該命令セットに限定されるものではないことは、以下の説明から明らかであろう。

本実施例では通常ＳＷのうち繰り返し実行されるソフトウェア部分（ループ）を、ＤＲＣに実行させる。これは、ＤＲＣドライバＳＷを通常ＳＷの実行中に自動生成するため、複数回実行されるソフトウェア部

分をＤＲＣに実行させるようにすることが効率的であると考えられるためである。第８図（Ａ）の例では、第４行目から第１８行目までのプログラムがその対象となる（第８図（Ｂ））。

本発明では、第８図（Ｂ）の通常ＳＷに基づき、ＤＲＣ書き換えデータとＤＲＣドライバＳＷ（第８図（Ｃ））とを生成する。ＤＲＣ書き換えデータは、ＤＲＣが第８図（Ｂ）のプログラムで実行される演算を実行するようにＤＲＣを設定するためのデータである。また、ＤＲＣドライバＳＷは、通常ＳＷのプログラム中のＤＲＣでは実行できない命令を実行し、汎用プロセッサからＤＲＣへデータを入力し、ＤＲＣで実行された結果を汎用プロセッサに戻すためのプログラムである。したがって、汎用プロセッサは、汎用レジスタＧＲと演算器ＡＬＵを用いて第８図（Ｂ）を実行する代わりに、第８図（Ｃ）を実行してＤＲＣを用いて演算を実行することにより、処理能力を向上させることができるのである。

第１図に本発明のＬＳＩチップ１００の構成を示す。ＬＳＩチップ１００は、バス・ステート・コントローラＢＳＣと、ダイレクト・メモリ・アクセス・コントローラＤＭＡＣと、オンチップ・メモリＯＣＭと、ＤＲＣドライバＳＷ格納メモリＤＳＭと、命令キャッシュＩＣＨと、データ・キャッシュＤＣＨと、キャッシュ制御ユニットＣＣＮと、ＣＰＵ１０１により構成される。ＣＰＵ１０１はＤＲＣ制御ユニット１０２と、命令バッファＩＢＦと、命令フェッチ・ユニットＩＦＵと、セレクタＳＥＬと、命令デコーダＩＤＣと、汎用レジスタＧＲと、演算器ＡＬＵと、ＤＲＣにより構成される。ＤＲＣ制御ユニット１０２は、書き換え判定ユニットＣＤＵと、ハードウェア／ソフトウェア生成ユニットＧＵ（以下、ＨＷ／ＳＷ生成ユニットと記述する）により構成される。

ＣＰＵ１０１が実行する命令は命令キャッシュＩＣＨに格納され、命令フェッチ・ユニットＩＦＵの命令ロード信号に従って命令キャッシュ

ＩＣＨ内に格納されている命令は命令バッファＩＢＦに転送される。同時に、書き換え判定ユニットＣＤＵは、命令キャッシュＩＣＨから命令バッファＩＢＦへ転送される命令を常に監視する。

書き換え判定ユニットＣＤＵは第８図（Ａ）の例でいえば、条件分岐命令ＢＦを検出し、それに基づきＤＲＣで実行する候補となるプログラムとして第８図（Ｂ）に示すソフトウェア部分を切り出し、格納する。ＤＲＣで実行するソフトウェア部分が決定すると、書き換え判定ユニットＣＤＵはＨＷ／ＳＷ生成ユニットＧＵに対して、ＤＲＣ書き換えデータの作成、ＤＲＣ書き換え、及びＤＲＣドライバＳＷの作成を指示する。

ＨＷ／ＳＷ生成ユニットＧＵは、切り出されたプログラムからＤＲＣ書き換えデータを生成し、ＤＲＣの書き換えを行う。また、書き換えたＤＲＣを利用するためのＤＲＣドライバＳＷを生成し、生成されたＤＲＣドライバＳＷはドライバＳＷ格納メモリＤＳＭに格納する。これらの処理が終了すると、ＨＷ／ＳＷ生成ユニットＧＵは書き換え判定ユニットＣＤＵに終了を通知するとともに、ＤＲＣドライバＳＷが格納される先頭アドレスを通知する。

なお、通常ＳＷはオンチップ・メモリＯＣＭや外部メモリ・チップＥＸＴＭに格納されているのが一般的である。

ＣＰＵ１０１によるプログラムの実行は以下のようになる。第８図の例により説明する。最初は第８図（Ａ）の通常ＳＷがそのまま実行される。ＨＷ／ＳＷ生成ユニットＧＵから終了通知を受けた後に、書き換え判定ユニットＣＤＵは、命令キャッシュＩＣＨから命令バッファＩＢＦへ転送される通常ＳＷの実行中の命令が、ＤＲＣドライバＳＷに置き換えられるプログラムの条件分岐命令（第８図（Ｂ）の第１５行、すなわち第８図（Ａ）の第１８行に相当する）まで処理が進んだことを確認すると、命令フェッチ・ユニットＩＦＵを介してセレクタＳＥＬを切り替

え、命令デコーダＩＤＣに入力される命令を命令バッファＩＢＦから判定ユニットＣＤＵに切り替える。次に、条件分岐命令の分岐先アドレスをＤＲＣドライバＳＷの格納される先頭アドレスに変換した命令を、セレクタＳＥＬを介して命令デコーダＩＤＣに出力する。次のＣＰＵクロック・サイクルでは、書き換え判定ユニットＣＤＵは命令フェッチ・ユニットＩＦＵを介してセレクタＳＥＬを切り替え、命令デコーダＩＤＣに入力される命令を判定ユニットＣＤＵから命令バッファＩＢＦに切り替える。これにより、以降はＤＲＣドライバＳＷが実行される。ＤＲＣドライバＳＷの最後の命令は、切り出した通常ＳＷの次の命令が格納されたアドレスへの無条件分岐命令（第８図（Ｃ）第１５行）であり、この命令の実行によって通常ＳＷの実行に戻る。

ここで、命令キャッシュＩＣＨへの命令の格納は、命令デコーダＩＤＣの実行を受けて、キャッシュ・コントローラＣＣＮによって行われる。キャッシュ・コントローラＣＣＮは、プロセッサ・バスＰＲＣＢ上のモジュールであるオンチップ・メモリＯＣＭと、ダイレクト・メモリ・アクセス・コントローラＤＭＡＣと、バス・コントローラＢＳＣ（外部メモリ・チップＥＸＴＭへのアクセスの場合）に加えてドライバＳＷ格納メモリＤＳＭに対してもメモリ・アクセスを実行可能に構成されている点が一つの特徴である。

第２図にＤＲＣドライバＳＷの生成及び実行のタイミングを示す。

まず通常ＳＷが実行され、命令フェッチ・ユニットＩＦＵ内のプログラム・カウンタに格納されている現在実行中の命令のアドレスより前のアドレスへの条件分岐が起こった場合（第８図（Ａ）第１８行）には、書き換え判定ユニットＣＤＵは、通常ＳＷ中にループがあると仮決定する。

書き換え判定ユニットＣＤＵは、以降の実行で命令キャッシュＩＣＨ

から命令バッファＩＢＦへロードされる命令、すなわち第８図（Ａ）第４行～第１８行の命令を取得して格納する。再度、条件分岐命令により格納された先頭の命令に戻った場合には、格納された命令はループを形成していると判定し、ＤＲＣの利用を本決定する。

ＨＷ／ＳＷ生成ユニットＧＵは、３回目のループでＤＲＣ書き換えデータの作成、ＤＲＣドライバＳＷの作成、およびＤＲＣの書き換えを行う。３回目のループにおいてＤＲＣの利用が可能になるので、ＣＰＵ１０１は、演算ユニットＡＬＵによる演算に代えて、第４回目以降のループはＤＲＣドライバＳＷを実行することによりＤＲＣにより演算を行うようにする。なお、ＤＲＣの書き換えが３回目のループ実行中に終わらない場合は、ＣＰＵ１０１はＤＲＣの書き換えが終了する時点を含むループの最後まで通常ＳＷを実行することになる。

次に、ＤＲＣの構造を第３図～第５図を用いて詳しく説明する。第３図はＤＲＣの内部構成を示している。入出力レジスタセルＩＯＲＣと、演算セルＣＣと、入出力レジスタセルＩＯＲＣへのデータ入力ポート２００と、レジスタ指定入力ポート２０１と、セル入力線２０３ａ、２０３ｂ、２０３ｃと、ＤＲＣのデータ出力ポート２０２と、各セルからのセル出力線２０４ａ、２０４ｂと、配線領域２０５と、配線プログラム素子２０６から構成される。配線プログラム素子２０６は、配線領域２０５中の配線同士の結線を決定するためのスイッチ素子とこれらスイッチ素子のオン／オフ状態を記憶するための図示しない記憶素子（例えば、ＳＲＡＭ、Ｆｌａｓｈメモリ等）により構成される。

ＤＲＣへのデータ入力時には、命令デコーダＩＤＣからレジスタ指定入力ポート２０１にレジスタ指定信号が入力され、１つの入出力レジスタ・セルＩＯＲＣが選択される。データはデータ入力ポート２００より入力され、選択された入出力レジスタ・セルＩＯＲＣのみに入力される。

また、ＤＲＣからのデータの出力時には、命令デコーダＩＤＣからレジスタ指定入力ポート２０１にレジスタ指定信号が入力される。これにより出力セレクタＯＳＥＬが切り換えられ、１つの入出力レジスタ・セルＩＯＲＣの出力が選択される。データは、セル出力線２０４ａより出力セレクタＯＳＥＬに入り、選択されたデータのみデータ出力ポート２０２より出力される。

第４図は、第３図の入出力レジスタ・セルＩＯＲＣの構成を示している。入出力レジスタ・セルＩＯＲＣは入力セレクタＩＳＥＬとセル・レジスタＣＲにより構成されている。入力セレクタＩＳＥＬは、レジスタ指定入力ポート２０１からの入力に応じて、配線からのセル入力線２０３ａまたはデータ入力ポート２００からの入力に切り換える。入力セレクタＩＳＥＬを介して入力されたデータは、セル・レジスタＣＲに保持される。このセル・レジスタＣＲはクロック入力３０１と同期して動作し、リセット入力３０２によってリセットされる。なお、第３図ではクロック及びリセットの配線は省略しているが、全ての入出力レジスタ・セルＩＯＲＣと演算セルＣＣとに接続されている。セル・レジスタＣＲに保持されたデータはセル出力線２０４ａより外部に出力される。

なお、本実施例ではデータは８ビット単位で入出力が行われているが、この大きさには限定されない。

第５図は、第３図の演算セルＣＣの構成を示している。演算セルＣＣは、セル演算ユニットＣＡＬＵと、フリップ・フロップＦＦと、演算プログラム素子４００から構成される。セル演算ユニットＣＡＬＵは、ＣＰＵ内のＡＬＵと同じ機能を持ち、演算プログラム素子４００によってどの演算機能を使用するかを設定する。セル演算ユニットＣＡＬＵの実行可能な論理和、論理積、排他的論理和といった論理演算、加減乗除といった算術演算、比較演算のうちから、演算プログラム素子４００はセ

ル演算ユニットＣＡＬＵが実行する演算を設定する。

このように、演算セルＣＣの演算プログラム素子４００の設定により、演算セルＣＣの演算内容を決定できる。また、配線プログラム素子２０６の設定により、いかなるデータを入出力レジスタ・セルＩＯＲＣ、演算セルＣＣに入力するのか、また入出力レジスタ・セルＩＯＲＣのデータまたは演算セルＣＣの演算結果をどこに出力するのかを設定することができる。このようにＤＲＣ書き換えデータは、演算プログラム素子４００の設定値と配線プログラム素子２０６の設定値とを含み、所望の演算を実行可能にする。

第６図に書き換え判定ユニットＣＤＵの構成を、第７図に書き換え判定ユニットＣＤＵの動作を示す。書き換え判定ユニットＣＤＵは、分岐アドレス格納バッファＢＡＢと、ループ・カウンタＬＣと、命令アドレス判定ユニットＩＡＤＵと、ＤＲＣ状態レジスタＤＳＲと、通常ＳＷ一時格納バッファＴＢＦと、分岐コントローラＢＣＬから構成される。さらに、ＤＲＣ状態レジスタＤＳＲは、ＨＷ／ＳＷ生成ユニットＧＵの状態を示す部分、ＤＲＣドライバＳＷへの分岐アドレスを格納する部分、およびＤＲＣドライバＳＷに置き換える通常ＳＷの次の命令があるアドレスを格納する部分の３つの部分で構成される。

以下、第７図に即して書き換え判定ユニットＣＤＵの動作を説明する。まず、命令キャッシュＩＣＨから命令バッファＩＢＦへ送られる命令を命令アドレス判定ユニットＩＡＤＵに取り込む（５００）。命令アドレス判定ユニットＩＡＤＵは、命令フェッチ・ユニットＩＦＵ中のプログラム・カウンタＰＣ内のアドレスを取り出し、その現在実行中の命令のアドレスがドライバＳＷ格納メモリＤＳＭのアドレス領域に該当するかどうかを判定する（５０１）。

該当する場合は現在ＤＲＣドライバＳＷを実行中であることを意味す

る。このとき、ループカウンタＬＣが０以外の値であれば（５０２）ＤＲＣドライバＳＷの第１行の命令（即ち第８図（Ｃ）の第１行のＭＯＶ命令）であるため、ループ・カウンタＬＣを０にリセットし、分岐コントローラＢＣＬは、セレクタＳＥＬに命令バッファＩＢＦ側に切り換えさせる。ループカウンタＬＣが０の場合には何もしない。ループ・カウンタＬＣには、通常ＳＷ中で連続して実行されるループの回数が保持されており、リセット信号によりリセットされる。

該当しない場合には現在通常ＳＷを実行中であることを意味するので、命令アドレス判定ユニットＩＡＤＵは、命令が条件分岐命令であるかどうかを判定する。条件分岐命令ではなく、かつこのときループ・カウンタＬＣの値が１であれば、第２回目のループを実行していることになる。そのため、通常ＳＷを取得するため（第２図を参照）、命令アドレス判定ユニットＩＡＤＵは命令を通常ＳＷ一時格納バッファＴＢＦに格納する。

ステップ５０５において、命令アドレス判定ユニットＩＡＤＵは、ＤＲＣ状態レジスタＤＳＲをチェックする。ＤＲＣ状態レジスタＤＳＲは、ＤＲＣの状態を示す第１レジスタと、ＤＲＣドライバＳＷへの分岐アドレス（例えば第８図（Ｃ）の第１行のＭＯＶ命令が格納されたアドレス）を格納する第２レジスタと、ＤＲＣドライバＳＷに切り換えられる通常ＳＷの次のアドレス（例えば第８図(A)のＬ００３：（第１９行））を格納する第３アドレスとから構成される。第１レジスタには「ＤＲＣ利用不可」、「ＤＲＣ利用準備中」、「ＤＲＣ利用準備完了」の３状態のいずれかが格納されている。ＣＰＵリセットを受けて「ＤＲＣ利用不可」の値に更新され、ＨＷ／ＳＷ生成ユニットＧＵによるＤＲＣ利用準備開始時に「ＤＲＣ利用準備中」の値に更新され、ＤＲＣ利用準備完了時に「ＤＲＣ利用準備完了」の値に更新される。ＤＲＣ利用準備完了時に第

２アドレスは更新され、ＧＵへの動作開始の通知時に第３アドレスは更新される。

ＤＲＣ状態レジスタＤＳＲの第１レジスタが「ＤＲＣ利用準備完了」の値であれば、分岐コントローラＢＣＬは、命令フェッチ・ユニットＩＦＵを介してセレクタＳＥＬをＤＲＣ制御ユニットＣＤＵに切り換えて、書き換え判定ユニットＣＤＵからの出力が命令デコーダＩＤＣに繋がるようにする。その後、分岐命令の分岐先アドレスをドライバＳＷ格納メモリＤＳＭ内のＤＲＣドライバＳＷの先頭アドレスに変更した分岐命令を送る。

ＤＲＣ状態レジスタＤＳＲの第１レジスタが「ＤＲＣ利用準備完了」の値以外であれば、ループの存在の判定を行う（第２図における仮決定処理）。まず、命令アドレス判定ユニットＩＡＤＵは、現在のプログラム・カウンタＰＣと分岐先アドレスとを比較する（５１３）。分岐先アドレスの方が大きい場合はループは存在しないので、ループ・カウンタＬＣを０とする（５１４）。プログラム・カウンタＰＣの方が大きい場合には、さらに命令アドレス判定ユニットＩＡＤＵは、分岐アドレス・バッファＢＡＢに格納された分岐先アドレスと、分岐命令の分岐先とを比較する。分岐アドレス・バッファＢＡＢは、分岐命令を実行した場合にその分岐先アドレスを上書き保存するバッファである。したがって、分岐先が分岐アドレス・バッファＢＡＢのアドレスに等しければ、ループの存在を決定し、ＤＲＣ書き換えデータ、ＤＲＣドライバＳＷの生成処理に移る。具体的には、ループ・カウンタに１を加算し（５０９）、その値が２であれば（５１０）、命令アドレス判定ユニットＩＡＤＵは、分岐命令を通常ＳＷ一時格納バッファＴＢＦに格納し、ＤＲＣ状態レジスタＤＳＲの第３レジスタに現在のＰＣの値を１つ進めた値（例えば図８（Ａ）のＬ００３：（第１９行）のアドレス）を代入して、ＨＷ／Ｓ

W生成ユニットＧＵに対してＤＲＣ利用準備の開始信号を送る（５１１）。

一方、分岐先が分岐アドレス・バッファＢＡＢのアドレスと異なっていれば、新たなループの存在の可能性がある。そこで、命令アドレス判定ユニットＩＡＤＵは、ループ・カウンタＬＣに１を代入し、通常ＳＷ一時格納バッファＴＢＦを一括クリアする。

上記動作を第８図のプログラム例に基づいて説明すると以下のようになる。

第１回目のループの実行では、第８図（Ａ）の第１行～第１７行の命令は、ループ・カウンタＬＣは０であり、第７図のフローでは何も実行されない。１８行目の条件分岐命令ＢＦが取り込まれると、ループ・カウンタＬＣを１にするとともに（５０８）、分岐先アドレス（第４行）が分岐アドレス・バッファＢＡＢに上書きされる。

第２回目のループの実行では、第５行～第１７行目の各命令は、ループ・カウンタＬＣが１なので、通常ＳＷ一時格納バッファＴＢＦに格納される（５０４）。第１８行目の条件分岐命令ＢＦが取り込まれると、その分岐先アドレスと分岐アドレス・バッファＢＡＢに格納されたアドレスとは一致するので（５０７）、ループ・カウンタＬＣを２として（５０９）、条件分岐命令ＢＦを通常ＳＷ一時格納バッファＴＢＦに送るとともに、ＨＷ／ＳＷ生成ユニットＧＵにＤＲＣ利用準備を開始させる（５１０，５１１）。

第３回目のループの実行では、第５行～第１７行目の各命令は、ループ・カウンタＬＣは２であるので、第７図のフローでは何も実行されない。第１８行目の条件分岐命令ＢＦが取り込まれ、ＤＲＣが利用可能な状態になっていれば、分岐命令の分岐先アドレスをＤＲＣドライバＳＷの先頭アドレス（第８図（Ｃ）第１行）に変更することで、以降のループの実行はＤＲＣを利用して行われる。

ＤＲＣドライバＳＷの実行では、ＪＭＰ命令（第８図（Ｃ）の第１５行）により通常ＳＷのＬ００３（第８図（Ａ）の第１９行）に分岐し、通常ＳＷの実行に復帰する。

次に、ＨＷ／ＳＷ生成ユニットＧＵの動作を第９図のフロー図を元に説明する。ＨＷ／ＳＷ生成ユニットＧＵは、書き換え判定ユニットＣＤＵからの命令を受け、通常ＳＷ一時格納バッファＴＢＦから通常ＳＷ（第８図（Ｂ））を取得するとともに、ＤＲＣ状態レジスタＤＳＲの第１レジスタには「ＤＲＣ利用準備中」の値を入力する（６００）。

ＨＷ／ＳＷ生成ユニットＧＵは、まず取得した通常ＳＷから第１０図に示すようなＣＤＦＧ(Control Data Flow Graph)を作成する（６０１）。ＣＤＦＧは、入力された通常ＳＷ中の各命令をノード（命令）で、命令オペランドのデータ依存関係をエッジ（矢印）で表したものである。なお、第１０図におけるブロックの括弧内の数字は第８図（Ｂ）の対応する行数を表している。

第１０図は、第８図（Ｂ）の通常ＳＷからそのままＣＤＦＧを作成したものである。依存関係には、「制御上の依存関係」と「データ上の依存関係」の２種類がある。「制御上の依存関係」とは、例えばＤＴ命令でレジスタＲ０の値を設定し、レジスタＲ０の値に応じて条件分岐命令ＢＦを実行するといった依存関係である。これは汎用プロセッサの命令セットに依存するものであり、あらかじめ依存関係を登録しておく必要がある。「データ上の依存関係」は、例えばＭＯＶ命令で転送されたデータを用いてＳＵＢで演算を行うといった依存関係である。したがって、プログラムの内容によって決定していく必要がある。オペランドに他の命令のオペランドと依存関係のないものを最上位に配置し、依存関係に応じてより下位に配置する。「データ上の依存関係」は具体的には以下のようにして決定する。

第８図（Ｂ）第２行～第４行、第１０行の命令のオペランドはそれより前の命令のオペランドに依存関係がないので、同列に位置する。「MOV @R6,R2」とは「レジスタＲ６の指示するアドレスに格納されたデータをレジスタＲ２に転送する」という意味である。すなわち、オペランドの右の値が@を含むＭＯＶ命令は外部からレジスタにデータを読み込むという命令であり、さらに@のついたオペランドは第８図（Ｂ）内で初の出現であるため、「データ上の依存関係」としては最上位に位置することになる。第５行の命令「MUL」は、レジスタＲ２，Ｒ３のデータを用いるため、第３行～第４行の命令と依存関係を有する。第７行の命令は「MUL」の実行結果はレジスタ MACL に入力されるため、レジスタ MACL のデータをレジスタＲ７に転送するためのものである。

以下のプログラムについても同様にして依存関係を決定するが、エッジの決定には同じレジスタ名であっても内容が更新されている場合があることに注意する必要がある。例えば、第８行の「SUB R7,R1」は「レジスタＲ７のデータとレジスタＲ１のデータの差をレジスタＲ１に格納する」という意味である。第８行の命令の実行によりデータが更新されているため、第９行の「MOV R1,@R5」は第８行の命令と依存関係はあるが、第６行の「MOV @R4,R1」とは依存関係が存在しない。

このように、着目する命令に対して、それ以前に実行された命令のうち、オペランドに共通のレジスタ名を有する命令と依存関係を設定することでＣＤＦＧが作成できる。ただし、共通のレジスタ名を有する命令が複数ある場合にはレジスタのデータが更新される可能性を考慮して、直前に実行された命令との間に依存関係を設定する。

第１０図のＣＤＦＧを縮約したのが、第１１図のＣＤＦＧである。縮約することでＤＲＣの回路構成を単純にし、ＤＲＣでの演算処理が高速に行えるといった効果がある。

第１の縮約の形態が７０１の２命令（第１０図）を８０１の１命令（第１１図）に縮約するものである。ＳＴＳ命令は一切の演算処理を行うことなくデータをレジスタ間で転送する命令である。このような転送は汎用プロセッサで処理する場合には命令セットの構成上必要となる命令ではあるものの、ＤＲＣで処理する場合には直接必要なデータの格納されたレジスタと演算セルとを接続するようにすればよいため、このような命令をＤＲＣの構成に反映させる必要はない。このような、データをレジスタ間で転送する命令はあらかじめ縮約対象とする命令として事前に登録しておくことで第１の縮約を実行できる。

第２の縮約の形態が７０２の４命令（第１０図）を８０２の３命令（第１１図）に縮約するものである。汎用プロセッサでは時系列で処理がされるため、レジスタ名が違っていてもその内容が同じであるということが生じうる。ＤＲＣで処理する場合には必要なデータの格納されたレジスタと演算セルとを接続するようにすればよいため、このようなレジスタ名の相違をＤＲＣの構成に反映させる必要はない。そのため、演算命令（第８図の例ではＭＵＬ命令、ＡＤＤ命令、ＳＵＢ命令）については、そのレジスタのデータの内容を比較し、レジスタが異なっていても同じデータが格納されている場合には縮約するようにする。例えば、第８行のＳＵＢ命令と第１１行のＡＤＤ命令のオペランドは、レジスタＲ７が共通し、レジスタＲ１、レジスタＲ３とで異なっている。しかしながら、それぞれデータ上の依存関係をたどるといずれのレジスタに格納された内容も「レジスタＲ４の指示するアドレスに格納されたデータ」であるため、上記のような縮約が可能になる。

次に、ＨＷ／ＳＷ生成ユニットＧＵは、ステップ６０１で作成した第１１図に示すＣＤＦＧの各ノードに対して、スケジューリング、すなわちハードウェア・リソースの制約を考慮した上で各ノードを実行するク

ロック・サイクルの割り当てを行う（６０２）。

このスケジューリングの一方法としてＡＬＡＰアルゴリズムとＡＳＡＰアルゴリズムを第１１図のＣＤＦＧに対して併用して適用した例を第１２図及び第１３図に示す。ＡＬＡＰ（As Late As Possible）アルゴリズムは同時に実行可能な命令を後から詰めて行くスケジューリングである。同時に実行できない命令の第一は相互に依存関係のある命令である。第二は、ＤＲＣの外部にアクセスする命令はそのサイクルに１つしか実行することができない。この結果が第１２図である。

さらに、第１２図のスケジューリングに対して、同時に実行可能な命令を前から詰めて行くＡＳＡＰ（As Soon As Possible）アルゴリズムを適用したものが第１３図である。できるだけ、命令を実行するサイクルを前に詰めることによって、処理時間に余裕が生まれる。

１サイクルに含まれる命令は１クロック・サイクルで実行することができ、サイクル１～８の８クロック・サイクルで全ての命令を実行することができる。

ＨＷ／ＳＷ生成ユニットＧＵは、第１３図に示すスケジューリングされたＣＤＦＧより、ＤＲＣ書き換えデータを生成する。具体的には、以下の通りである。ノード部分を入出力レジスタ・セルＩＯＲＣ及び演算セルＣＣに割り当てる。ＭＯＶ命令については入出力レジスタ・セルＩＯＲＣを割り当て、演算命令については演算セルＣＣを割り当てる。演算命令の内容に応じて、演算セルの演算内容の設定を行う演算プログラム素子４００の設定データを作成する。また、ＣＤＦＧのエッジはデータの流れを示しているので、ＣＤＦＧのエッジの接続関係にしたがって、入出力レジスタ・セルＩＯＲＣ及び演算セルＣＣが相互に配線されるように、配線プログラム素子２０６の設定データを作成する（６０３）。この演算プログラム素子４００の設定データ及び配線プログラム素子２

０６の設定データがＤＲＣ書き換えデータである。

ＨＷ／ＳＷ生成ユニットＧＵは、ＤＲＣ書き換えデータにしたがってＤＲＣの演算プログラム素子３０３と配線プログラム素子２０６を書き換える（６０４）。これと並行してＤＲＣドライバＳＷの作成を行う（６０５）。

ＤＲＣドライバＳＷの作成方法について説明する。第１３図のＣＤＦＧから第８図（Ｃ）に示すＤＲＣドライバＳＷを生成する。

まず、汎用レジスタＧＲに格納されているデータをＤＲＣの入出力レジスタ・セルＩＯＲＣに移動させる必要がある。例えば、第８図（Ｂ）第３行の「MOV @R6, R3」という命令は「汎用レジスタＲ６の指示するアドレスに格納されたデータを汎用レジスタＲ３に転送する」命令である。この命令をＤＲＣで実行するためには、まず汎用レジスタＲ６のデータをＤＲＣ中に移動させておく必要がある。したがって、第８図（Ｃ）第１行「MOV R6, dR6」という命令が設けられる。この命令は「汎用レジスタＲ６に格納されたデータ（アドレス）をＤＲＣの入出力レジスタ・セルｄＲ６に転送する」という意味である。このような命令が第８図（Ｃ）の第１行～第３行に設けられている。

その後、第１３図の各ノードのうち、ＤＲＣ以外で実行される命令がＤＲＣドライバＳＷに設けられる。第８図（Ｃ）の第５行～第１２行の命令はそれぞれ第１３図のサイクル１～８に含まれるノードに対応している。

第８図（Ｃ）第５行「MOV @dR6, dR3」は「入出力レジスタ・セルｄＲ６の指示するアドレスに格納されたデータを入出力レジスタ・セルｄＲ３に転送する」命令であり、これは第１３図のサイクル１の「MOV @R6, R3」に対応している。以下同様であるが、例えばサイクル３では「MOV @R4, R1」、「MUL」の２つのノードが存在しており、後者はＤＲＣが実行する

命令であるから、ＤＲＣドライバＳＷには現れない。また、サイクル６は「ADD」というＤＲＣが実行する命令しか存在していない。その場合には、空きサイクルが発生しているとして、ノー・オペレーション命令（ＮＯＰ）を追加する。

なお、演算命令についてはＤＲＣで実行させるために、演算に必要なデータは汎用レジスタＧＲではなく、ＤＲＣの入出力レジスタ・セルＩＯＲＣに格納することになる。そのため、演算実行後の値を汎用レジスタＧＲに戻す必要がある。したがって、第８図（Ｃ）の第１３行と第１４行の命令が設けられている。例えば、「MOV dR5, R5」は「入出力レジスタ・セルｄＲ５に格納されたデータを汎用レジスタＲ５に転送する」命令である。

最後に、通常ＳＷの実行に復帰するために、通常ＳＷにおける条件分岐命令の次のアドレスへの無条件分岐命令（ＪＭＰ）（第８図（Ｃ）第１５行）を設ける。この無条件分岐命令の分岐先はＤＲＣ状態レジスタの第３レジスタに格納されたアドレスとする。

このように、第１３図のスケジューリングされたＣＤＦＧを元に第８図（Ｃ）のＤＲＣドライバＳＷが生成できる。キャッシュのミスや割り込みなどの外的要因を排除して考えると、もとの通常ＳＷが一回のループの実行に１４サイクル必要だったのに対して、ループのコア部分（第８図（Ｃ）の第５行～第１２行）の８サイクルで実行できる。

作成されたＤＲＣドライバＳＷは、ドライバＳＷ格納メモリＤＳＭに格納される。格納する場所は、初回ならばドライバＳＷ格納メモリＤＳＭの先頭アドレスとし、以降は前回書き込んだ部分の後ろに書き込む。書き込む領域が十分でない場合には、再びドライバＳＷ格納メモリＤＳＭの先頭アドレスから書き込むようにする。

ステップ６０５まで終了すると、ＨＷ／ＳＷ生成ユニットＧＵはＤＲ

Ｃ状態レジスタＤＳＲの第１レジスタに「ＤＲＣ利用準備完了」の値を書き込み、ＤＲＣ状態レジスタＤＳＲの第２レジスタにステップ６０５で作成したＤＲＣドライバＳＷの格納されたメモリの先頭アドレスを書き込む（６０６）。

以上に述べたフローにしたがってＨＷ／ＳＷ生成ユニットＧＵを動作させることにより、プログラム実行時に自動的にＤＲＣ書き換えデータ作成および書き換えと、ＤＲＣドライバＳＷの作成を行うことができる。

第１４図を用いて第１図の構成の変形例を説明する。この実施例は、第１図に示した第１の実施例のＤＲＣ、書き換え判定ユニットＣＤＵ、ＨＷ／ＳＷ生成ユニットＧＵを、ＣＰＵ１０１から分離した例である。具体的には、ＤＲＣをプロセッサ・バスＰＲＣＢに接続し、セレクタＳＥＬの制御を書き換え判定ユニットＣＤＵが直接制御する。また、命令バッファＩＢＦをセレクタＳＥＬの後に配置している。

第１図の構成との動作の違いは、ＤＲＣの入出力レジスタ・セルＩＯＲＣへのアクセスがアドレスを指定して行われることである。例えば、第８図（Ｃ）のＤＲＣドライバＳＷの第１行目［MOV R6 dR6］などのようなＤＲＣの入出力レジスタ・セルＩＯＲＣにアクセスしている命令は、ＤＲＣが直接プロセッサ・バスＰＲＣＢに接続しているために、入出力レジスタ・セルＩＯＲＣのアドレスが直接指定ではなく、間接指定となる。例えば、［MOV R6 @R12］のようなアドレス指定になる。この命令は、汎用レジスタＲ６のデータを汎用レジスタＲ１２で指示される入出力レジスタ・セルＩＯＲＣに転送することを意味している。

この変形例では、ＤＲＣをＣＰＵの外部に配置することで、第１図の構成に比べてＤＲＣのサイズを大きくすることが可能である。また、ＤＲＣと、書き換え判定ユニットＣＤＵと、ＨＷ／ＳＷ生成ユニットＧＵがＣＰＵ１０１と分離されているために、設計の変更が容易である。さ

らに、ＣＰＵ以外のモジュールがＤＲＣを利用することも可能である。例えば、ダイレクト・メモリ・アクセス・コントローラＤＭＡＣにより、ＤＲＣ上の入出力レジスタセルＩＯＲＣにアクセスしてＤＲＣを利用することができる。この場合には、ＤＲＣとＣＰＵ１０１とで異なる処理を同時に実行させることもできる。

第１５図を用いて第１図の構成のさらに別の変形例を説明する。この構成例では、第１図に示したＤＲＣ制御ユニット１０２の機能を、ＤＲＣ制御専用プロセッサＤＣＰで実現している点が異なる。本構成例では、ＤＲＣ制御ユニット１０２の機能をＤＲＣ制御専用プロセッサＤＣＰで実装するため、その内部の書き換え判定ユニットＣＤＵ、およびＨＷ／ＳＷ生成ユニットＧＵの動作をソフトウェアで実現する。従って、ＤＲＣ制御専用プロセッサＤＣＰのソフトウェアを入れ替えるだけで容易にＤＲＣ制御ユニット１０２を更新することができ、本技術を利用したＬＳＩチップ製造後においても本ＬＳＩの性能を向上させることが可能となる。もちろん、第１４図の構成と第１５図の構成を組み合わせて実施することも可能である。

さらに、本発明は種々の変形が可能である。例えば、実行する通常ＳＷを予備的に実行しておき、ＤＲＣ書き換えデータ及びＤＲＣドライバＳＷを予め登録しておくようにしてもよい。この場合には通常ＳＷの実行中にＤＲＣ書き換えデータ及びＤＲＣドライバＳＷの生成処理が不要になるため、第２回目のループからＤＲＣを用いた演算処理が行える。

本発明の構成により、プロセッサとＤＲＣを搭載する半導体装置において、ソフトウェアからＤＲＣ書き換えデータとＤＲＣドライバＳＷとを自動生成することが可能になる。かかる構成により、ＤＲＣを利用する場合にもＤＲＣにあわせた専用のプログラムを記述する必要がなくなり、ソフトウェアの互換性を保つことができる。このように、ソフトウ

ェアの互換性を維持することで、既存のソフトウェア資産を流用しつつ少なくとも同じ命令セットで機能するプロセッサであれば同じソフトウェアを用いることができる。

5 産業の利用可能性

本発明は、特に複数回繰り返されるソフトウェア（例えば、ループを形成しているソフトウェア）に有効であり、このようなループは画像処理や音声処理でよく現れる処理である。

請 求 の 範 囲

1．演算命令を含むソフトウェアを実行する半導体装置であって、

レジスタと、

演算器と、

複数の演算セルと複数のレジスタセルとを含み、上記演算セルが実行する演算種と上記複数の演算セル及び上記複数のレジスタセル間の配線接続とを設定可能な演算回路と、

上記ソフトウェアに基づき、上記演算セルの上記演算種及び上記配線接続を設定する設定データと、上記演算回路を用いて上記ソフトウェアと等価な処理を行うためのドライバ・ソフトウェアとを生成する制御回路とを有する半導体装置。

2．請求項1において、

上記制御回路は、上記レジスタと上記演算器とを用いて上記ソフトウェアを実行している期間中に、上記設定データ及び上記ドライバ・ソフトウェアを生成するように構成された半導体装置。

3．請求項2において、

上記ソフトウェアの処理はn回繰り返され、

第1回から第i回（i＜n）までの処理は、上記レジスタと上記演算器とを用いて上記ソフトウェアを実行することによって行われ、

第i＋1回から第n回までの処理は、上記演算回路を用いて上記ドライバ・ソフトウェアを実行することによって行われる半導体装置。

4．請求項1において、

上記ドライバ・ソフトウェアは少なくとも上記レジスタから上記演算回路の上記レジスタセルへのデータ転送命令と、上記演算回路の上記レジスタセルから上記レジスタへのデータ転送命令とを含む半導体装置。

5．請求項1において、

制御回路は、上記設定データ及び上記ドライバ・ソフトウェアを生成するソフトウェアを実行することにより、上記設定データ及び上記ドライバ・ソフトウェアを生成する半導体装置。

6．請求項１において、

上記演算回路はバスに接続された半導体装置。

7．請求項１において、

上記ソフトウェアを実行するのに要するクロックサイクル数は上記ドライバ・ソフトウェアを実行するのに要するクロックサイクル数よりも小さい半導体装置。

8．演算命令を含むソフトウェアを実行する半導体装置であって、

レジスタと、

演算器と、

複数の演算セルと複数のレジスタセルとを含み、上記演算セルが実行する演算種と上記複数の演算セル及び上記複数のレジスタセル間の配線接続とを設定可能な演算回路と、

上記ソフトウェアを格納する第１メモリ領域と、

上記演算回路を用いて上記ソフトウェアと等価な処理を行うためのドライバ・ソフトウェアを格納する第２メモリ領域と、

実行するソフトウェアを制御する制御回路とを有し、

上記ソフトウェアの処理はｎ回繰り返され、

第１回から第ｉ回（ｉ＜ｎ）までの処理は、上記レジスタと上記演算器とを用いて上記第１メモリ領域から読み出された上記ソフトウェアを実行することによって行われ、

上記制御回路は上記第ｉ回の処理を受けて、実行するソフトウェアを上記ドライバ・ソフトウェアに切り換えることにより、第ｉ＋１回から第ｎ回までの処理は、上記演算回路を用いて上記第２メモリ領域から読

み出された上記ドライバ・ソフトウェアを実行することによって行われる半導体装置。

９．請求項８において、

上記制御回路は、上記ソフトウェアに基づき、上記演算セルの上記演算種及び上記配線接続を設定する設定データと、上記演算回路を用いて上記ソフトウェアと等価な処理を行うためのドライバ・ソフトウェアとを生成する半導体装置。

１０．請求項８において、

上記制御回路は、演算回路の上記演算種及び上記配線接続を設定する設定データを有し、

上記ソフトウェアを実行している期間中に、上記演算回路は、上記設定データにより、上記演算セルの上記演算種及び上記配線接続を設定する半導体装置。

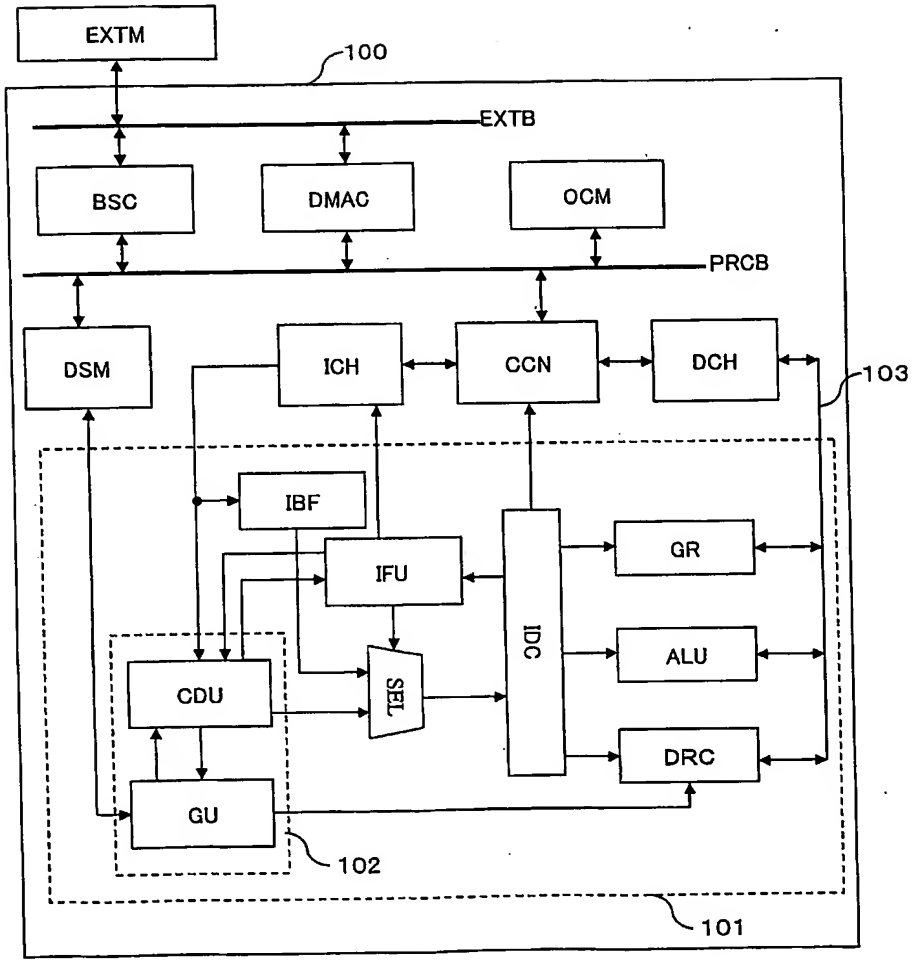
１１．請求項８において、

上記ドライバ・ソフトウェアは少なくとも上記レジスタから上記演算回路の上記レジスタセルへのデータ転送命令と、上記演算回路の上記レジスタセルから上記レジスタへのデータ転送命令とを含む半導体装置。

１２．請求項８において、

上記ドライバ・ソフトウェアを実行するのに要するクロックサイクル数は上記ソフトウェアを実行するのに要するクロックサイクル数よりも小さい半導体装置。

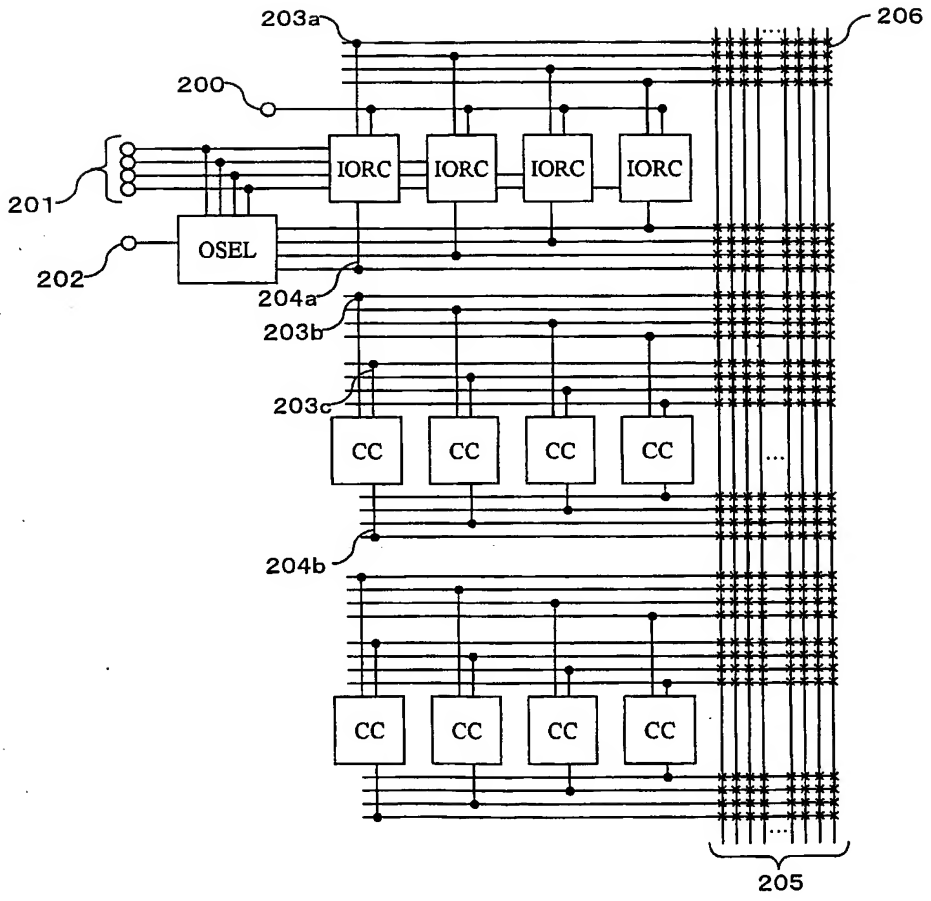
第1図


EXTM
100
EXTB
BSC
DMAC
OCM
PRCB
DSM
ICH
CCN
DCH
103
IBF
IFU
GR
IDC
ALU
CDU
SEL
DRC
GU
102
101

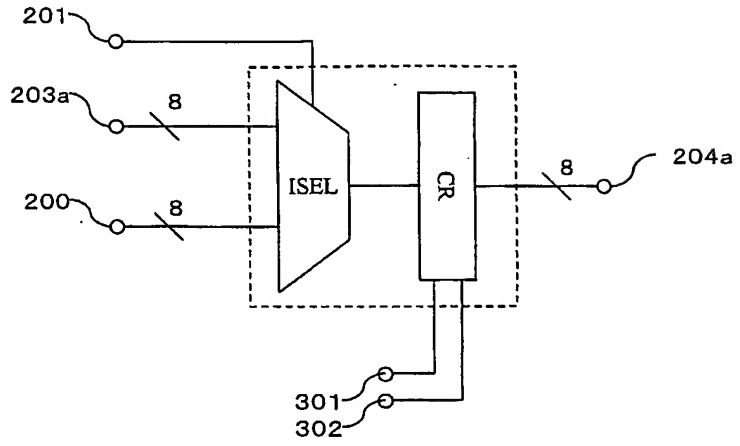
第2図


ループ回数
DRC制御ユニット102の動作
実行されるソフトウェア
1回目
仮決定
通常SW
2回目
通常SW取得
本決定
通常SW
3回目
DRC書き換え
通常SW
4回目
DRCドライバSW
5～n回目
DRCドライバSW

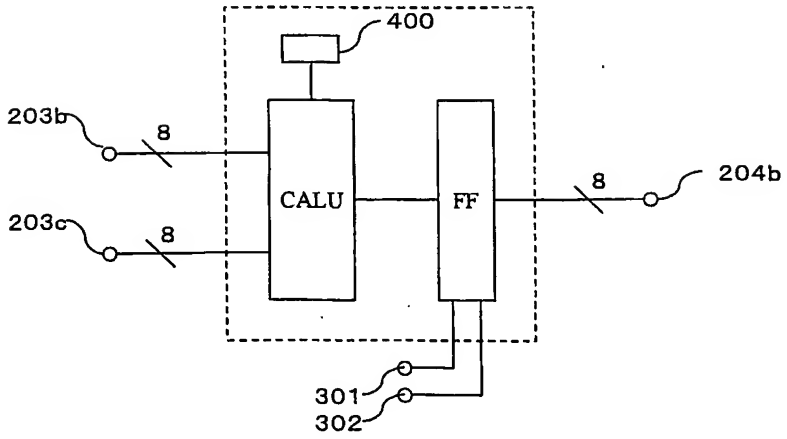
第3図


203a
206
200
IORC
IORC
IORC
IORC
201
OSEL
202
204a
203b
203c
CC
CC
CC
CC
204b
CC
CC
CC
CC
205

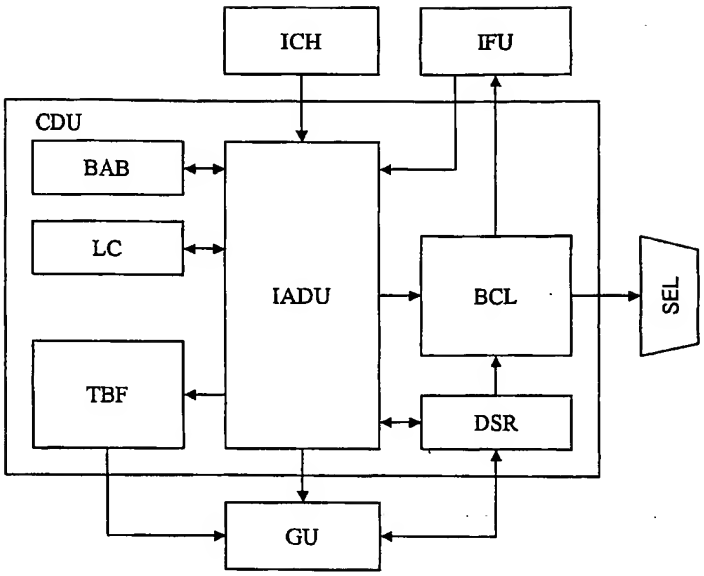
第4図


201
203a
8
200
8
ISEL
CR
8
204a
301
302

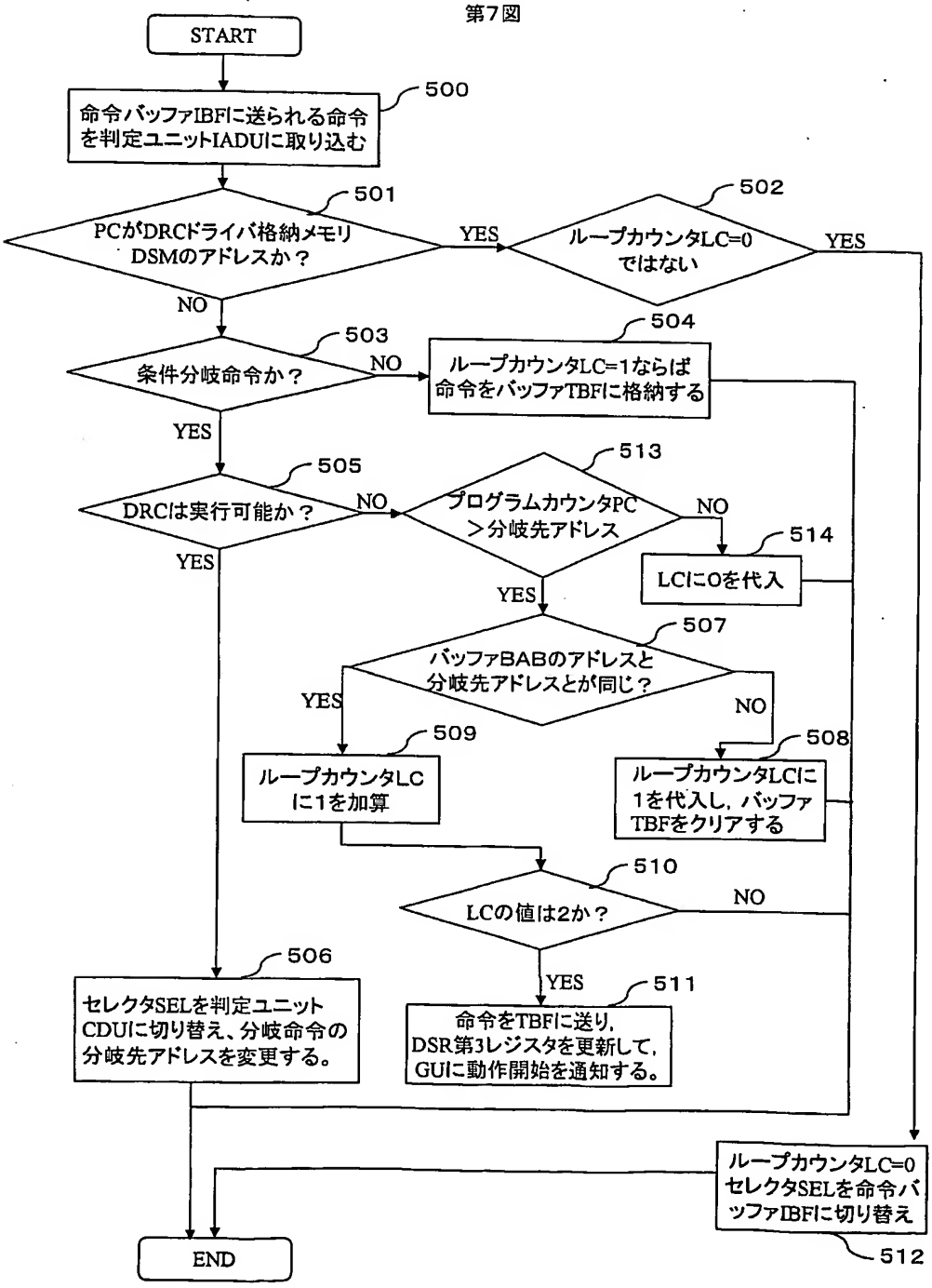
第5図


400
203b
8
203c
8
CALU
FF
8
204b
301
302

第6図



第7図


START
命令バッファIBFに送られる命令を判定ユニットIADUに取り込む
500
PCがDRCドライバ格納メモリDSMのアドレスか？
501
YES
ループカウンタLC=0ではない
502
YES
NO
条件分岐命令か？
503
NO
ループカウンタLC=1ならば命令をバッファTBFに格納する
504
YES
DRCは実行可能か？
505
NO
プログラムカウンタPC＞分岐先アドレス
513
NO
LCに0を代入
514
YES
YES
バッファBABのアドレスと分岐先アドレスとが同じ？
507
YES
NO
ループカウンタLCに1を加算
509
ループカウンタLCに1を代入し，バッファTBFをクリアする
508
LCの値は2か？
510
NO
YES
セレクタSELを判定ユニットCDUに切り替え、分岐命令の分岐先アドレスを変更する。
506
命令をTBFに送り，DSR第3レジスタを更新して，GUに動作開始を通知する。
511
ループカウンタLC=0セレクタSELを命令バッファIBFに切り替え
512
END

第8図

(A)

```
1  L001:
2    STS     MACL,@-R15
3    MOV     #8,R0
4  L002:
5    DT      R0
6    MOV     @R6,R3
7    MOV     @R5,R2
8    MUL     R3,R2
9    MOV     @R4,R1
10   STS     MACL,R7
11   SUB     R7,R1
12   MOV     R1,@R5
13   MOV     @R4,R3
14   ADD     R3,R7
15   MOV     R7,@R4
16   ADD     #4,R4
17   ADD     #4,R5
18   BF      L002
19 L003:
20   NOP
21   LDS     @R15+,MACL
22   RTS
23   NOP
```

(B)

```
1  L002:
2    DT      R0
3    MOV     @R6,R3
4    MOV     @R5,R2
5    MUL     R3,R2
6    MOV     @R4,R1
7    STS     MACL,R7
8    SUB     R7,R1
9    MOV     R1,@R5
10   MOV     @R4,R3
11   ADD     R3,R7
12   MOV     R7,@R4
13   ADD     #4,R4
14   ADD     #4,R5
15   BF      L002
```

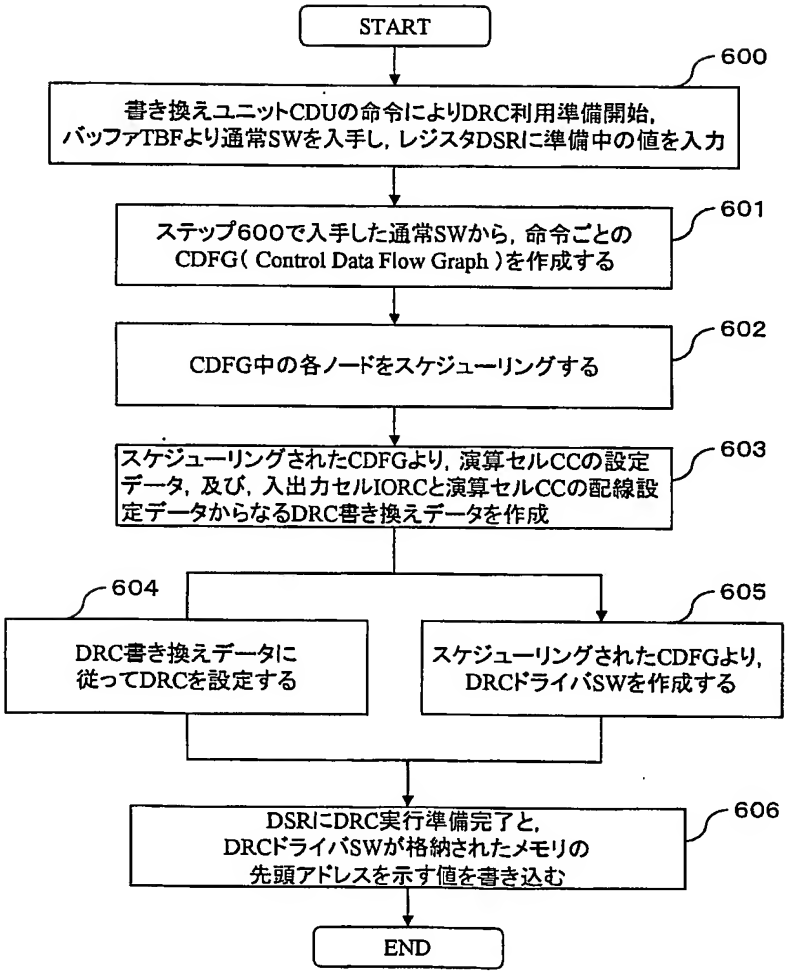
(C)

```
1    MOV     R6,dR6
2    MOV     R5,dR5
3    MOV     R4,dR4
4  L002a:
5    MOV     @dR6,dR3
6    MOV     @dR5,dR2
7    MOV     @dR4,dR1
8    DT      R0
9    MOV     dR1,@dR5
10   MOV     dR7,@dR4
11   NOP
12   BF      L002a
13   MOV     dR5,R5
14   MOV     dR4,R4
15   JMP     L003
```

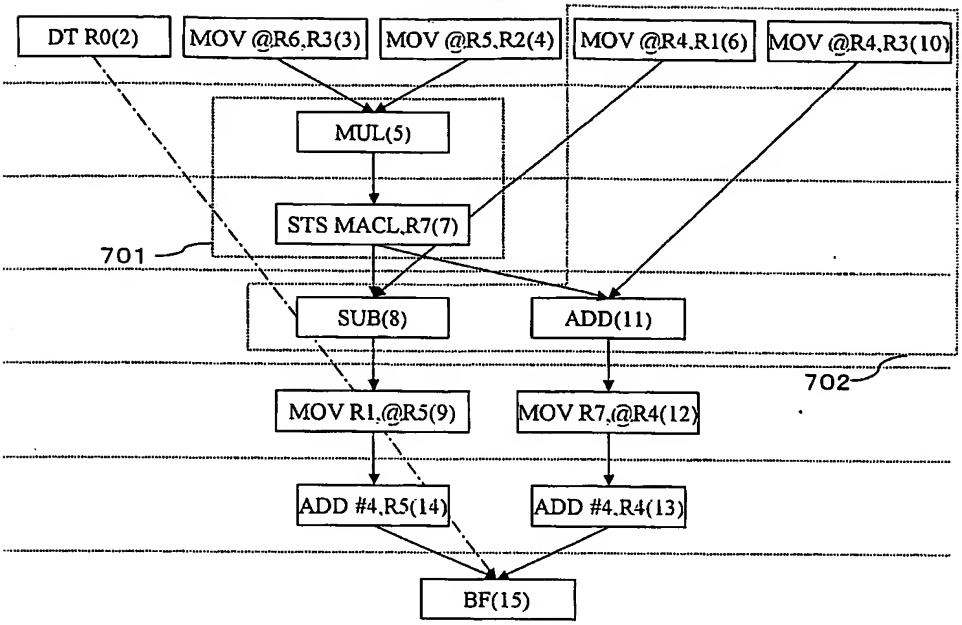
(D)命令の意味

MOV	データ転送命令
ADD	加算命令
SUB	減算命令
MUL	積算命令
DT	デクリメント＋条件セット命令
BF	条件分岐命令
JMP	無条件分岐命令
NOP	No Operation
LDS	システムレジスタロード
STS	システムレジスタストア
RTS	リターン命令
R*	汎用レジスタ*
dR*	IORC*

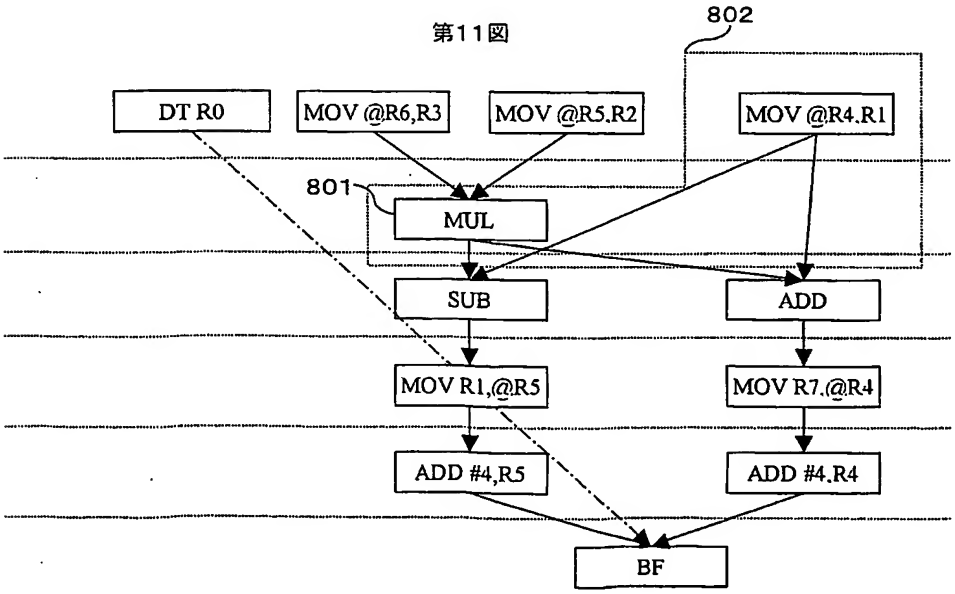
第9図


START
600
書き換えユニットCDUの命令によりDRC利用準備開始，
バッファTBFより通常SWを入手し，レジスタDSRに準備中の値を入力
601
ステップ600で入手した通常SWから，命令ごとの
CDFG(Control Data Flow Graph)を作成する
602
CDFG中の各ノードをスケジューリングする
603
スケジューリングされたCDFGより，演算セルCCの設定
データ，及び，入出力セルIORCと演算セルCCの配線設
定データからなるDRC書き換えデータを作成
604
DRC書き換えデータに
従ってDRCを設定する
605
スケジューリングされたCDFGより，
DRCドライバSWを作成する
606
DSRにDRC実行準備完了と，
DRCドライバSWが格納されたメモリの
先頭アドレスを示す値を書き込む
END

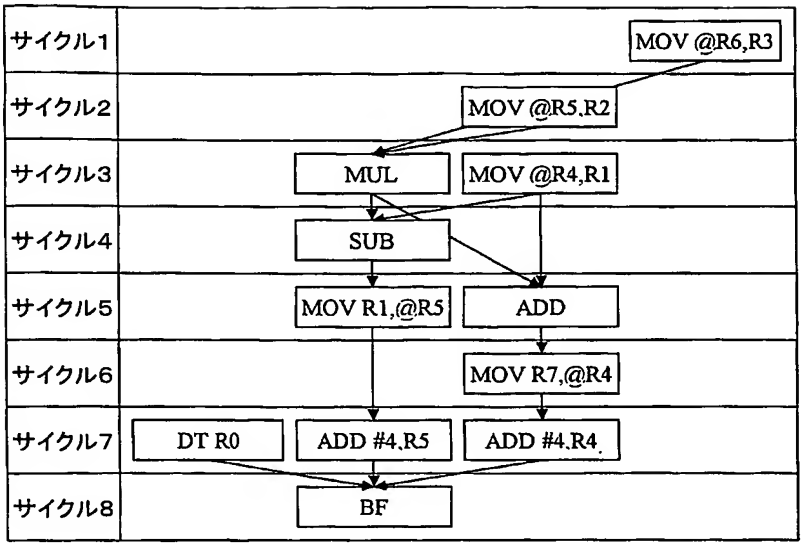
第10図


DT R0(2)
MOV @R6,R3(3)
MOV @R5,R2(4)
MOV @R4,R1(6)
MOV @R4,R3(10)
MUL(5)
STS MACL,R7(7)
701
SUB(8)
ADD(11)
702
MOV R1,@R5(9)
MOV R7,@R4(12)
ADD #4,R5(14)
ADD #4,R4(13)
BF(15)

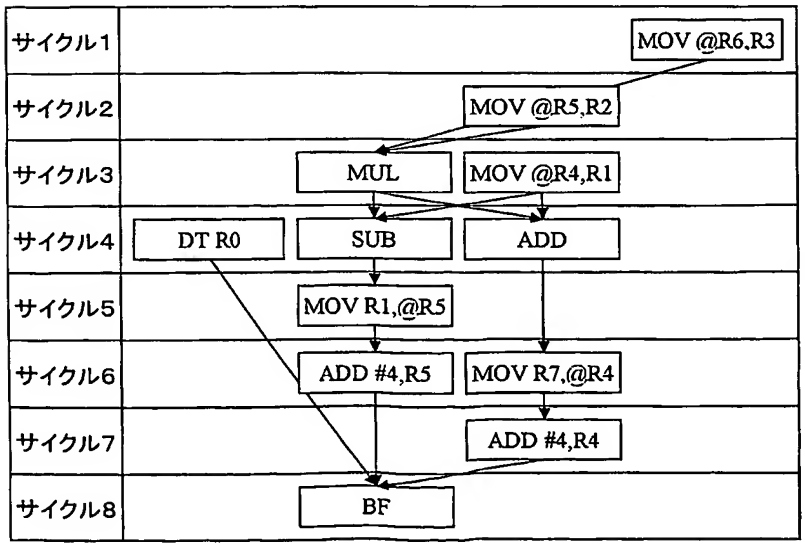
第11図


802
DT R0
MOV @R6,R3
MOV @R5,R2
MOV @R4,R1
801
MUL
SUB
ADD
MOV R1,@R5
MOV R7,@R4
ADD #4,R5
ADD #4,R4
BF

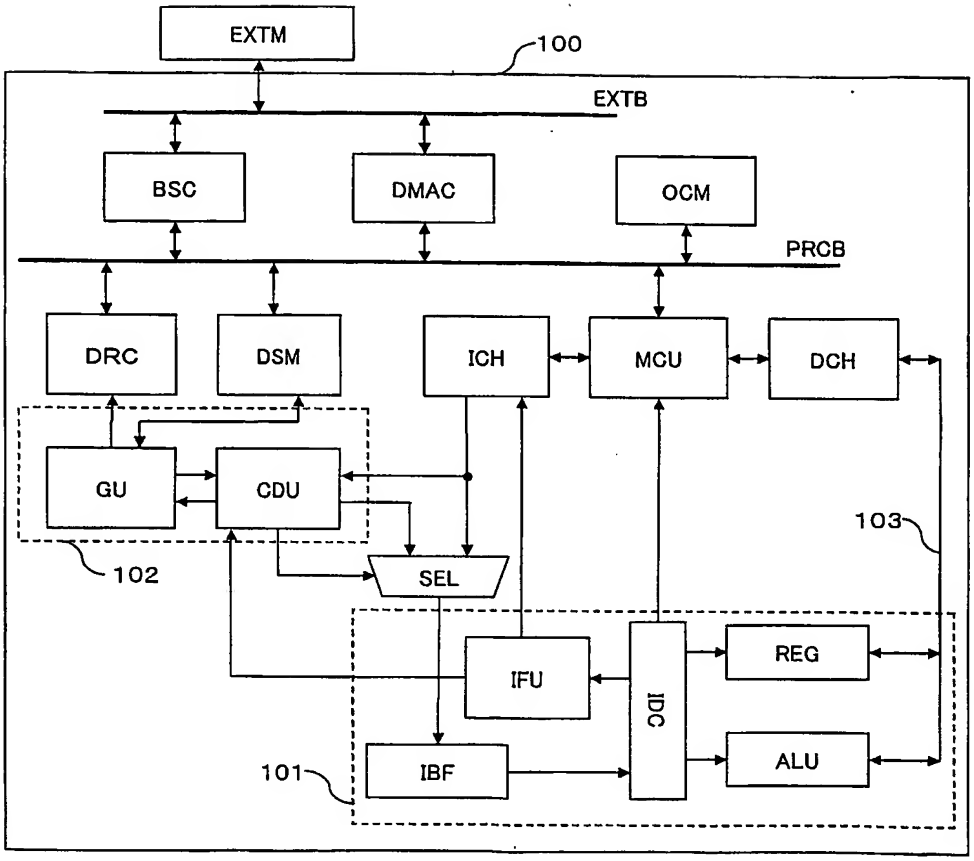
第12図


サイクル1
MOV @R6,R3
サイクル2
MOV @R5,R2
サイクル3
MUL
MOV @R4,R1
サイクル4
SUB
サイクル5
MOV R1,@R5
ADD
サイクル6
MOV R7,@R4
サイクル7
DT R0
ADD #4,R5
ADD #4,R4
サイクル8
BF

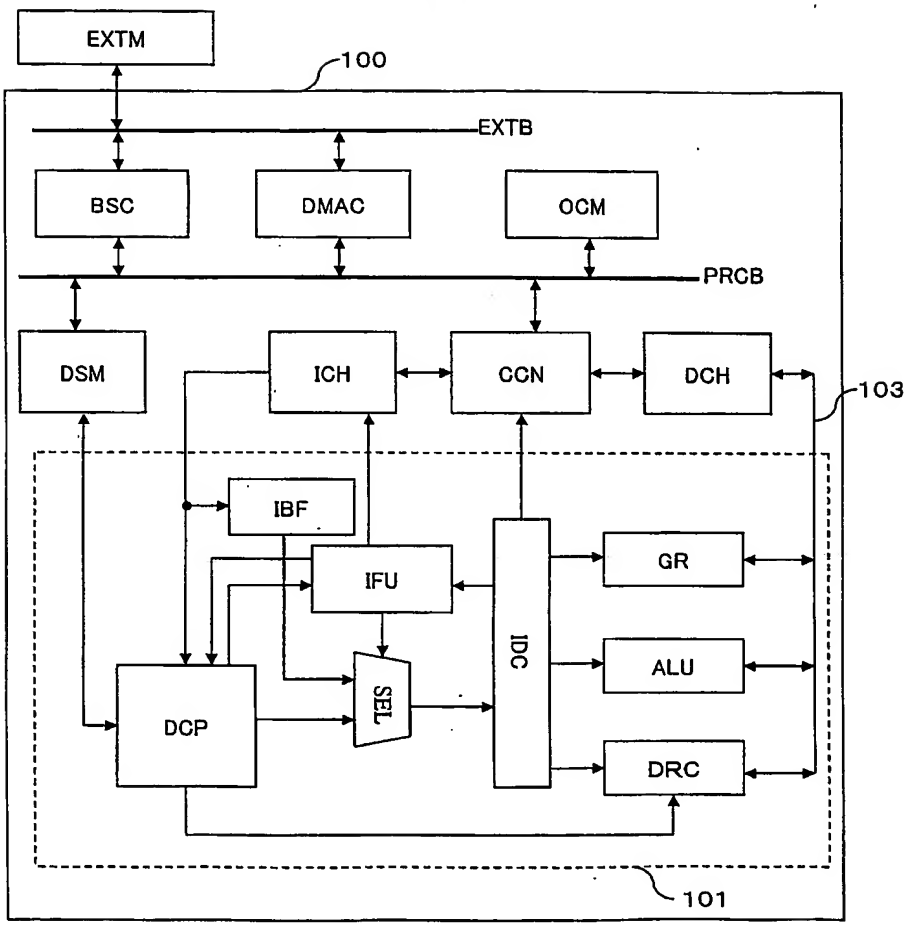
第13図


サイクル1
MOV @R6,R3
サイクル2
MOV @R5,R2
サイクル3
MUL
MOV @R4,R1
サイクル4
DT R0
SUB
ADD
サイクル5
MOV R1,@R5
サイクル6
ADD #4,R5
MOV R7,@R4
サイクル7
ADD #4,R4
サイクル8
BF

第14図


EXTM
100
EXTB
BSC
DMAC
OCM
PRCB
DRC
DSM
ICH
MCU
DCH
GU
CDU
103
102
SEL
IFU
IDC
REG
ALU
101
IBF

第15図


EXTM
100
EXTB
BSC
DMAC
OCM
PRCB
DSM
ICH
CCN
DCH
103
IBF
IFU
GR
IDC
ALU
DCP
SEL
DRC
101